NASA-CR-192739

NAGW-1333

CIRSSE DOCUMENT # 40

TECHNICAL REPORTS

GRANT
IN [illegible]
153763
P- 18

(NASA-CR-192739) STRUCTURAL FORMULATION OF PLAN GENERATION FOR INTELLIGENT MACHINES (Rensselaer Polytechnic Inst.) 18 p

N93-71632

Unclas

Z9/63 0153763

Center for Intelligent Robotic Systems for Space Exploration

Rensselaer Polytechnic Institute
Troy, New York 12180-3590

STRUCTURAL FORMULATION OF PLAN GENERATION FOR INTELLIGENT MACHINES

By:

F. Wang
G.N. Saridis

Department of Electrical, Computer and Systems Engineering
Department of Mechanical Engineering, Aeronautical Engineering & Mechanics
Rensselaer Polytechnic Institute
Troy, New York 12180-3590

September 1989

CIRSSE Document #40

STRUCTURAL FORMULATION OF PLAN GENERATION FOR INTELLIGENT MACHINES

FEIYUE WANG AND GEORGE N. SARIDIS

NASA Center for Intelligent Robotics Systems for Space Exploration
Robotics and Automation Laboratories
Department of Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute
Troy, New York 12180

Abstract - The issue of structural formulation of plan generation for Intelligent Machines is investigated from the perspective of Artificial Intelligence. Using a *relational database* to represent the knowledge base and a *condition-event net* to model the planning process, it is shown that plans generated by the formulation developed in this paper are Petri net languages and that any planning strategy can be achieved by a *supervisory planner*. The objective of such a structural formulation is to outline a formal framework in term of Mathematical Logic for plan generation of Intelligent Machines in such a way that it will serve as the foundation of analytical plan generation at different levels of the hierarchical structure of Intelligent Machines. In other words, this framework will play the role of "domain space" upon which various analytical design approaches such as the probablistic method, the neural network computing, etc, can be integrated to specify the plan generation in both functional aspect and computational aspect.

Keywords: *Intelligent Machine, Relational database, Condition-event net, Supervisory planner, Petri net Language.*

1. INTRODUCTION

The theory of Intelligent Machines is the result of the intersection of the three major disciplines of Artificial Intelligence, Operation Research, and Control Theory [Saridis 1980]. The structure of Intelligent Machines is defined to be the structure of hierarchically intelligent control systems, composed of three levels hierarchically ordered according to the principle of **Increasing Precision with Decreasing Intelligence**, namely: the *organization level*, the *coordination level*, and the *execution level* [Saridis 1979, 83, 85, 88a]. An Intelligent Machine can be thought as a machine system which can generate automatically the programms for the given tasks and then compile and execute them. It is the responsibility of plan generation to assign an Intelligent Machine the ability of automatic programm generation.

The plan generation of intelligent systems has been one of the most challenging tasks of Artificial Intelligence from the very beginning. Since Green's resolution-based planner [Green 1969] and Fikes-Nilsson's STRIPS system [Fikes 1971], considerable amount of efforts has been made to develop the new plan generation theory and to improve the efficiency of the existing plan generation methods [Warren 1974, Sacerdoti 1974 and 1977, Tate 1977, Stefik 1981a and 1981b, Wilkins 1984-85, Chapman 1985, Pednault 1986, and Lifschitz 1987, etc.].

The central issues in plan generation of intelligent systems are the state representation, the system architecture, and the planning strategy. The state (or situation) representation specifies the way of describing the knowledge about the environment and the internal system states, and the way of changing the environment. For most plan generation systems, the first order predicate calculus and the likes are used in the representation. The system architecture defines on a global level the framework employed to carry out the plan generation tasks. Some architectures, such as tropistic agent, knowledge-level agent, and functionally accurate and cooperative system, have been proposed for single agent and multiple agents [Lesser 1981, Newell 1982, Genesereth 1987]. Heuristics has played the key role in the planning strategy. Most of the planning strategies in the existing plan generation systems have used heuristic methods, like the depth or breadth-first search, the best-first search, A^* algorithm, means-ends analysis, and their combinations [Fikes 1971, Pearl 1984, Ligeza 1985].

For the analytical design of Intelligent Machines, however, analytical plan generation is mostly desired. To this end, we try in this paper to formalize the representation, the architecture, and the strategy of plan generation in terms of some well-formed mathematical concepts developed in mathematical logical and discrete event theory. The relational database is used to represent the knowledge of the Intelligent Machine. The condition event net is introduced to model the planning process. The supervisory control method in the theory of discrete event system is employed as a general mechanism to specify the planning strategy. The possibility of implementing the analytical methods like probabilistic and neural network computing within the proposed the plan generation framework is discussed.

2. KNOWLEDG BASE REPRESENTATION

The machine knowledge, or simply, the knowledge, of an Intelligent machine reflects primarily in three aspects, that is, the knowledge about the environment, the knowledge about its own ability, and the knowledge about the causal relationship between the environment and its ability.

The level of abstraction of the knowledge of an Intelligent Machine varies in the different levels of the Machine, leading to different patterns of combination of the three aspects of the knowledge. In the lowest level of Intelligent Machine, i.e., the execution level, these three aspects of the knowledge are inextricably integrated in the form of real-time executable procedures and there is no point to discuss them separately. In the higher levels of Intelligent Machine, i.e., the organization and coordination levels, however, a clear distinguishtion among these three aspects is essential for the structural formulation of plan generation since the knowledge there is basically descriptive one.

The *knowledge about environment* includes the generic or the specific objects recognized, the properties perceived, and the relationships among the perceived properties known by the Intelligent Machine. For example, An object designated as *container* may be recognized by an Intelligent Machine in the organization level; a container may be *empty* or *full,* properties perceived about container by the Intelligent Machine, and either empty or full, but not both, is true at any time for a container, a relation between the two properties known by the Intelligent Machine. Note that a container in the organization level late may turn out to be a glass or a cup in the coordination level.

We find that an elegant representation of such knowledge can be provided by a relational database [Reiter 1984, Yang 1986] (or more generally, a deductive database [Lloyd 1987]). A relational database is a triple **DB=(L, T, IC)** where **L=(Δ, W)** is a *relational language* with **Δ=(V, O, P)** as its *alphabet* and **W** as the *well-formed formulas* (wffs) on Δ. Specifically,

$\mathbf{V} = \{x, y, z, ...\}$ is the infinite set of *variables.*

$\mathbf{O} = \{o_1, o_2, ..., o_s\}$ is the finite set of *constants* representing the name of objects recognized by the Intelligent Machine.

$\mathbf{P} = \{p_1, p_2, ..., p_m\}$ is the finite set of *predicates* representing the name of properties perceived by the Intelligent Machine.

$\mathbf{T} \subseteq \mathbf{W}$ is a *(generalized) relational theory* , that is, **T** contains exclusively the *domain closure axioms*, the *unique name axioms*, the *axioms for the equality,* and *the completion axioms* for each predicates but equality [Reiter 1984]. **T** describes what an Intelligent Machine knows about the properties of objects and their relationships.

$\mathbf{IC} \subseteq \mathbf{W}$ is a set of wffs, called *integrity constraints* . The integrity constraints corresponds to the so-called static integrity constraints or state laws in [Nicolas 1978]. Such constraints, specifying the facts which are not changeable under the actions of the Intelligent Machine, are meant to be satisfied by any state of the database.

It should be noted that we consider a relational database from the point of proof-theoretic view, not the point of model-theoretic view. As been pointed out in by [Reiter 1984], from the point of proof-theoretic view the relational databases can be perceived as special kinds of first order theories, therefore one can generalized those those theories in order to provide solution to a variety of questions. For our purpose, the most important thing is that the use of relational databases from the proof-theoretic view opens ways to apply various methods in the theory of mathematical logic in the plan generation of Intelligent Machine. We will, however, not abandon the model-theoretic view absolutely. As it can be seen later that the model-theoretic view will lead to a condition-event net formalism for the planing processes naturally.

The *knowledge about its own ability* of an Intelligent Machine is characterized by the actions which can be taken by the Intelligent Machine. Each action corresponds to an action routine (i.e., a sequence of actions in the lower level) whose execution causes the Intelligent Machine to make certain actions which would physically change the its state and the environment. An *action set*, $A=\{a_1(\mathbf{x}_1), a_2(\mathbf{x}_2), ..., a_n(\mathbf{x}_n)\}$, is used to represent these actions, where $\mathbf{x}$ in $a(\mathbf{x})$ is the *parameter sequence* of action $a(\mathbf{x})$. For example, grasp(x) is an action representing the routine to make a Robot to grasp the object x. An *event* associated with an action $a(\mathbf{x})$ is an instance of applying the action to some objects, i.e., $a(\mathbf{o})$, where $\mathbf{o}$ is a tuple of the constants of the L 's alphabet. Events are the basic elements from which a plan is constructed. The notation $E(A)=\{a(\mathbf{o}) | a(\mathbf{x}) \in A$ and $\mathbf{o}$ is a tuple of the constants$\}$ represents the set of all events associated with the actions in A. A *plan* s of the Intelligent Machine is defined to be any string on E(A), i.e., $s \in E(A)^*$.

The *knowledge about tthe causal relationship between the environment and its ability* is expressed by the constraints imposed on the actions and the effects of applying the actions.The constraints and effects can also be considered as the pre- and post-conditions of actions and be described by a mapping F from actions to the pairs of wffs, i.e., $F: A \rightarrow W \times W$, where $F(a(\mathbf{x}))=(W_1(\mathbf{x}), W_2(\mathbf{x}))$, $W_1(\mathbf{x})=W_{11} \wedge W_{12} \wedge ... \wedge W_{1g}$, $W_2(\mathbf{x})=W_{21} \wedge W_{22} \wedge ... \wedge W_{2h}$, W_{ij} are *atomic formulas* and the only free variables in W_{ij} are those in $\mathbf{x}$. For example, F(grasp(x))= (Near(Robot,x)∧HandEmpty(Robot), Hold(Robot,x)), tells that in order to grasp the object x, the Robot must be near x and with its hand empty, the result of the action is that the Robot holds x.

An action $a(\mathbf{x})$ is *applicable* in the database **DB** iff its precondition is true in **DB**, i.e., when $F(a(\mathbf{x}))=(W_1(\mathbf{x}), W_2(\mathbf{x}))$ and there exists a constant tuple **d** such that $\mathbf{T}\Rightarrow W_1(\mathbf{d})$, i.e., $W_1(\mathbf{d})$ is true can be derived from **T**. Once the action is applied with respect to objects **d**, $W_1(\mathbf{d})$ will be retracted from and $W_2(\mathbf{d})$ will be asserted into the theory **T** respectively, resulting in a corresponding modification in the completion axioms for the related predicates in **T**. In this way the theory **T** of **DB** is changed to a new theory **T'**, denoted as $\mathbf{T}'=(\mathbf{T}, a(\mathbf{d}))$, correspondingly, **DB** to $\mathbf{DB}'=(\mathbf{DB}, a(\mathbf{d}))$. A plan is *applicable* in **DB** iff all the actions in the plan are applicable in the database resulted from the last action application. Let $L_a(\mathbf{A})$ denote the set of all plans on **A** applicable in **DB**, called the *compatible* plans on **A**. For any plan s in $L_a(\mathbf{A})$, (**T**, s) and (**DB**, s) are used to represent the final theory and database resulted from applying the plan s on **DB**, respectively.

The above description defines formally a represention for the knowledge of an Intelligent Machine, we will call the triple **KB=(DB, A, F)** as the *knowledge base* of the Intelligent Machine.

To describe the interaction between an Intelligent Machine and its users, the *command language* is introduced. The *command language* is the language used to specify the tasks to be accomplished by the Intelligent Machine. We define the command language in a similar way as that for the quarry language of the relational database [Reiter 78]. Specifically, a command for $\mathbf{L}=(\Delta, \mathbf{W})$ is any express of the form $<\mathbf{x} \mid W(\mathbf{x})>$ where

1. **x** denotes the sequence $x_1, x_2, ..., x_n$, and x_i's are variables of Δ;
2. $W(\mathbf{x}) \in \mathbf{W}$ and the only free variables in $W(\mathbf{x})$ are those in **x**.

Let **DB=(L, T, IC)** be a relational database, then a command for **L** is said to be applicable to **DB**. The command language of **DB** is defined to be the set of all commands applicable to **DB**. Since, obviously, a command applicable to **DB** is also applicable to (**DB**, s) for any $s \in L_a(\mathbf{A})$, the command language for a knowledge base **KB=(DB, A, F)** of an Intelligent Machine, denoted as L_{com}, is the the command language of **DB**.

Assuming $c=<x \mid W(x)>\in L_{com}$ is a command, c is said to be *executable* by the Intelligent Machine if and only if there exist a string $s\in L_a(A)$ and a constant tuple **o** such that $(\mathbf{T}, s)\Rightarrow W(\mathbf{o})$. There may be more than one plan can be used to execute the command $c = <x \mid W(x)>$, let $L(c,A)=\{s \mid s\in L_a(A)$ and $(\mathbf{T}, s)\Rightarrow W(\mathbf{o})$ for some constant tuple $\mathbf{o}\}$, then L(c,A) is the set of all plans which can be used to accomplish the task c. We call $L_o(A)= \cup_{c\in L_{com}} L(c,A)$ the set of *compatible and complete* plans of the Intelligent Machine. All the tasks can be accomplished by the Intelligent Machine is $U=\{c \mid c\in L_{com}$ is executable by Intelligent Machine$\}$. Note that U might be an infinite set.

3. CONDITION EVENT NET

To formalize the planning processes, we first introduce the condition-event net developed in [Wang 1988a].

Let D be a set. A *term* on D is either an element of D or a variable on D. Similar to the concept of "*universe of discourse*" introduced by [Zadeh 1971], a *world* of D is a subset of the set which contains D and which is generated from D by a finite application of operation +(union), ×(direct product), and $\mathcal{P}$(power set). A *place* p over D is a variable on $\mathcal{P}(W)$, where W is a world of D. W is called the *domain* of p, denoted by D(p)=W. If p_1 and p_2 are two places with the same domain, then $p_1 + p_2$ and $p_1 - p_2$ are defined to the union and difference sets of p_1 and p_2, respectively.

A *statement* s on D is a pair s=(p, t),where p is a place over D and t is a term on D(p). s is called a *ground* statement iff t is an element, otherwise called a *variable* statement. A ground statement s=(p, t) is *true* iff $t\in p$, otherwise s is *false*. Let P be a set of places over D, a *condition* C on P over D is a collection of statements $C=\{s_1, s_2, ..., s_k\}$, $s_i=(p_i,t_i)$, $p_i\in P$, with $\{p_1, p_2, ..., p_k\}$ as its *place set* and $\{t_1, t_2, ..., t_k\}$ its *term set*. A condition C is *ground* iff all the statements in C are ground. A ground condition C is *true* iff all the statements in C are true, otherwise C is *false*. We use C(P, D) to denote the set of all conditions on P over D.

Definition : A *condition-event* (C-E) net structure, CE, is a 4-tuple, CE={O, A, P, F} where

$O = \{o_1, o_2, ..., o_s\}$ is the finite set of *object* s, $s \geq 0$;

$A = \{a_1, a_2, ..., a_m\}$ is the finite set of *actions* , $m \geq 0$;

$P = \{p_1, p_2, ..., p_n\}$ is the finite set of *places*, $n \geq 0$;

$F: A \rightarrow C(P, D) \times C(P, D)$ is a *condition function*, a mapping from actions to condition pairs on P over O.

Let $F(a_i)=(C_{i1}, C_{i2})$ be the condition pair associated with the action a_i, we call C_{i1} the *precondition* of a_i, denoted by $Pre(a_i)=C_{i1}$, and C_{i2} the *postcondition* of a_i, denoted by $Post(a_i)=C_{i2}$. A place p_j is an *input place* of a_i iff it belongs to the place set of $Pre(a_i)$; p_j is an *output place* of a_i iff it belongs to the place set of $Post(a_i)$. The *parameter set* of action a_i is the set of variables occurring in the term set of $Pre(a_i)$ or that of $Post(a_i)$.

As for Petri net, we can introduced the concepts such as the *C-E net graph, the C-E net state space, the marking of C-E net, the execution rule* (i.e., *enabled, fire,* etc.), the *next-state function*, etc., in similar way. For example, the *state* of a C-E net is defined to the values of its places. Obviously, a C-E net reduces to a Petri net when there is just one object and all the places have the same domain. It also can be shown for any C-E net there exists an equivalent, but usually quite large, Petri net. Therefore, even they have the same expression power, it is more convenience to use C-E net as model for the planning process, especially the places in C-E net is closely related with the predicates as can be seen below.

To formalize the planning process of building plans using the actions in a selected subset of the action set A by a C-E net, it is necessary to reinterpret the predicate in term of place. In the standard interpretation for the first order logic theory, this is invalid since a predicate there is interpreted as a fixed subset of some product space of a designated domain [Mendelson 1979]. In the present problem, however, the application of actions on a database changes its theory and, consequently, changes the subsets represented by the corresponding predicates, since it has been proved in [Reiter 1984] there is one-to-one corresponding between the theories and models for the relational database. Therefore it is quite naturally to describe the predicates as the variable subsets, i.e., as the place, correspondingly, the atomic formulae as the statements, the conjunctive wffs of atomic formulae as the conditions, for the plan generation problem. With these illustrations, given a subset B of the action set A, a C-E net can be constructed as follow:

$CE(B)=(O, B, P, F_b)$ where

O is the constant set of the database of the Intelligent Machine.

P is the finite predicate set of the database of the Intelligent Machine.

F_b is the restriction of F of the Intelligent Machine on the subset B.

CE(B) is called the *C-E net associated with the action subset B*. The planning process of building plans using the actions in B now can be considered as the firing process of actions of the C-E net CE(B), i.e., the executing of actions on CE(B). Let δ be the next-state function and m_0 be the initial *state* of the C-E net CE(B) which can be determined from the database **DB**. Clearly, $L_a(\mathbf{B})$ is the set of all firing sequences of actions in B from the initial state, i.e., $L_a(\mathbf{B})=\{s \mid s\in E(\mathbf{B})^* \text{ and } \delta(s,m_0) \text{ is defined}\}$. For convenience, if $m=\delta(s,m_0)$, we write $(\mathbf{T}, m)\equiv(\mathbf{T}, s)$.

4. THE MECHANISM OF PLAN GENERATION

To build plans for a given task described by a command, the first step to be carried out is to select a set of appropriate actions for the accomplishment of the task. This can be represented by a mapping γ from commands to the subsets of the action set A, i.e.

$$\gamma: L_{com} \rightarrow \mathcal{P}(A)$$

the mapping γ can expressed by a binary vectors where a 1-component indicates the corresponding action is selected (therefore, is active) and a 0-component indicates the corresponding action is not (therefore, is inactive). Note that for the most of researches in Artificial Intelligence, $\gamma(c)=A$ for all $c \in L_{com}$. For the cases of large number of actions, this may lead to poor efficiency. Generally, it might be very difficult to determine the γmapping, however, the probabilistic version of γ may the computed and learned efficiently by means of neural network as indicated in [Saridis 1988b].

For a given command $c=<x \mid W(x)>$, a marked C-E net, $CE(c,B_c)$, i.e., a C-E net with a initial state and a goal set can be defined as:

$CE(c,B_c)=(m_0, CE(B_c), m_0, G_c)$ where

$CE(B_c)$ is the C-E net associated with $B_c=\gamma(c)$;

m_0 is the initial state of $CE(B_c)$ which can be obtained from the database **DB**;

$G_c=\{m \mid m$ is a state of $CE(B_c)$ and $(T, m) \Rightarrow W(\mathbf{o})$ for some constant tuple $\mathbf{o}\}$.

$CE(c,B_c)$ is called the *C-E net associated with the command c.* Since C-E net and Petri net are equivalent, it is easy to show that a C-E net with a initial state and a goal set is equivalent to a labelled Petri net with identity labelling function% [Peterson 1981]. Therefore, L(c), the set of all plans on $\gamma(c)$ which achieve the task c, *is a L-type Petri net lauguage*, since G_c is always a finite set of states for a relational database. This result clarifies the lauguage complexity of plans generated with the formulation described.

No result has been given about the lauguage property of plans as for as we know. For an Intelligent Machine which can execute very general tasks, a compiling system is required to process the plans generated, therefore the lauguage property of plans is useful and imperative in the case. The formal model for the coordination level of Intelligent Machines developed in [Wang 1988b] is capable of compiling the Petri net language in a distributed and concurrent way. Note

% Note that, with the labelling function, the action with more than one preconditions can be described by the different actions with the same label.

that few applications have been found for Petri net lauguage, this result reveals the importance of Petri net language in the plan generation problem.

A general mechanism to implement the planning strategy using the set-up described now can be specified. Since any planning strategy is just some special method used to find (heuristically or analytically) one or more string(s) of the language L(c) for the givens task c. Recalling the supervisory control theory of discrete event systems [Ramadge 1982 and 85], It says for a given language L(G) with G as the generator (an automation, possibly an infinite one), any of its sub-language K can be specified by a supervisor S such that K = L(S/G), where S/G is the language generator of S supervising G. Basing on this fact, Ramadge-Wonham's supervisory control method can be used as the general mechanism of plan generation. To this end, for the C-E net $CE(c,B_c)$, we define a *supervisory planner*, S(c), to be a pair

$S(c)=(S_c, \varphi)$ where

$S_c = (\Sigma, X, \xi, x_0, X_m)$ is an *automaton* with

$S=E(\gamma(c))$ as the input alphabet;

X as the state set;

$\xi:\Sigma \times X \rightarrow X$ as the transition function;

x_0 as the initial state;

X_m as the set of final states;

$\varphi:\Sigma \times X \rightarrow \{0,1\}$ is a *plan law*.

Taking $CE(c,B_c)$ as a plan generator whose output is the firing sequence of actions, S(c) is considered to be driven externally by the stream of output actions of $CE(c,B_c)$; while in turn, with S in state x, action events of $CE(c,B_c)$ are subject to the plan law φ. If φ(a(d), x) = 0 then a(d) is 'disabled' (prohibited from firing), while if φ(a(d), x) = 1 then a(d) is 'permitted' (but may not be enabled). The enabling condition of $CE(c,B_c)$ under supervision of S(c) therefore is replaced by: an action event is enabled iff it is both enabled in $CE(c,B_c)$ and permitted by S(c). As in [Ramadge

1982], we use $S(c)/CE(c,B_c)$ to denote the resulting planning structure, and δ_φ to denote the modified next-state function.

Let $L(S(c)/CE(c,B_c))=\{s|\ s\in E(\gamma(c))^*,\ \delta_\varphi(s,m_0)$ and $\xi(s,x_0)$ are defined$\}$ and $L_m(S(c)/CE(c,B_c)) =\{s|\ s\in E(\gamma(c))^*,\ \delta_\varphi(s,m_0)\in G_c$ and $\xi(s,x_0)\in X_m\}$. We have the following important theorem:

Theorem: Let K be any subset of L(c), then there exists a supervisory planner S(c) such that $K= L_m(S(c)/CE(c,B_c))$.

The proof of the theorem is similar to that of the corresponding theorem in [Ramadge 1982]. This theorem claims that any planning strategy can be specified by constructing a special supervisory planner. Various results, such as modular feedback logic [Ramadge 1986], modular synthesis [Wonham 1988], decentralized control [Lin 1986] and other reduction methods [Vaz 1986] in the supervisory control theory seem to be useful in the structural construction of the planners.

Like the mapping γ, it may be very difficult, if not impossible, to construct the plan law φ. In the research works of plan generation in Artificial Intelligence, heuristics is usually used in the determination of the plan law. Since φ is also can be represented as a binary vectors, some probablistic fashion of φ may be computed and learned through neural network as for the mapping γ. Note that there is no direct way to use the neural network to generate plans, since the number of nodes needed to represent the plans is undefined, varying in an infinite range. Therefore the supervisory planner approach may provide an indirect way to compute the plans using neural network.

As the knowledge base KB representing the knowledge of Intelligent Machines, the mapping γ and the supervisory planners representing the intelligence of Intelligent Machines in the plan generation.

5. DISCUSSION AND CONCLUSION

A structural formulation of plan generation is developed with the objective of setting up a formal framework in term of well-formed mathematical models for plan generation of Intelligent Machines such that it will serve as the foundation of analytical plan generation at different levels of the hierarchical structure of Intelligent Machines. The relational database from the proof-theoretic view is used to represent the knowledge base of Intelligent Machines. Using the condition-event net as the formalism for planning process, it is shown that plans generated by the formulation are Petri net languages, an important result for the plan compiling and coordinating system. A supervisory planning approach is suggested and it is also proved that any planning strategy can be achieved by a supervisory planner. The possible ways of implementing the analytical methods within the proposed the plan generation framework is discussed in the context.

Further investigations of integrating the analytical methods into the plan structure are required. Once such integration is achieved, some analytical performance measures like entropies can be calculated for the different functional parts of the plan generation.

Also, generally, a task described by a command c can not expressed directly as the consequence of applying the actions in the database, however, it may be proved basing on the action consequence. Therefore a *theorem prover* is required in general for plan generation. There are several well-developed theorem provers available now. A plan generation system may even have more than one theorem provers, treating different type of tasks with different theorem prover.

References

Chapman, D. (1985) *Planning for Conjunctive Goal*, TR:83-85 (MS Thesis), Artificial Intelligence Lab., MIT, Cambridge, MA.

Fikes, R.E. and Nilsson, N.J. (1971) STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, *Artificial Intelligence*, Vol.2, pp189-208.

Genesereth, M.R. and Nilsson, N.J. (1987) *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann Publishers Inc., Los Altos, C.A.

Green, C. (1969), Application of Theorem Proving to Problem Solving, *Proc. 1st IJCAI*, Washington, D.C., pp219-239.

Lesser. V.R. and Corkill, D.D. (1981) Functionally Accurate, Cooperative Distributed Systems, *IEEE Trans. Syst., Man, and Cybn.*, Vol. SMC-11, No.1 pp81-96.

Lifschitz. V. (1987) On the Semantics of STRIPS, in *Reasoning about Actions and Plans*, Ed. by Georgeff, M. and Lansky, A., Morgan Kaufmann Publishers Inc., Los Altos, C.A.

Ligeza,A. (1985) Backward versus Forward Plan Generation, *Proc. IFAC Symposium on Robot Control*, ed. by Basanez, L., Ferrate, G., and Saridis, G.N., pp337-342.

Lin, F. and Wonham, W.M. (1986) Decentralized Supervisory Control of Discrete Event Systems, Systems Control Group Report #8612, Dept. EE, University of Toronto, Toronto, Canada.

Lloyd, J.W. (1987) *Foundations of Logic Programming*, Second, Extended Edition, Spring-Verlag.

Mendelson, E (1979) *Introduction to Mathematical Logic* , 2nd Edition, Van Nostrand, Princeton, N.J.

Newell, A. (1982) The Knowledge Level, *Artificial Intelligence*, Vol.18, No., pp87-127.

Nicolas, J.M. and Yaqdanian, K. (1978) Integrity Checking in Deductive Databases, in *Logic and Databases*, ed. by Gallaire, H. and Minker, J., pp325-344, Plenum Press, New York, N.Y.

Pearl, J. (1984) *Heuristics*, Addison-Wesley, Reading, MA.

Pednault, E. (1986) *Toward a Mathematical Theory of Plan Synthesis*, Ph.D. Dissertation, Dept. EE, Standford University, Standford, CA.

Peterson, J.P. (1981) *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Englewood Cliffs, N.J.

Ramadge, P.J. and Wonham, W.M. (1982) Supervision of Discrete Event Processes, *Proc. 21st IEEE Conf. on Decision and Control*, pp1228-1229.

Ramadge, P.J. and Wonham, W.M. (1985) Supervisory Control of A Class of Discrete Event Processes, Systems Control Group Report #8515, Dept. EE, University of Toronto, Toronto, Canada.

Ramadge, P.J. and Wonham, W.M. (1985) Modular Feedback Logic for Discrete Event Systems, Systems Control Group Report #8614, Dept. EE, University of Toronto, Toronto, Canada.

Reiter, R. (1978) Deductive Question-Answering on Relational Databases, in *Logic and Databases,* ed. by Gallaire, H. and Minker, J., pp149-177, Plenum Press, New York, N.Y.

Reiter, R. (1984) Towards a Logical Reconstruction of Relational Database Theory, in *On Conceptual Modelling,* ed. by Brodie, M. L., Mylopoulos, J. and Schmidt, J.W., pp191-233, Spring-Verlag.

Sacerdoti, E.D. (1974) Planning in a Hierarchy of Abstraction Spaces, *Artificial Intelligence,* Vol.5, No.2, pp115-135.

Sacerdoti, E.D. (1977) *A Structure for Plans and Behavior,* New York: Elsevier.

Saridis, G.N. (1979) Toward Realization of Intelligent Control, *Proc. IEEE,* Vol.67.

Saridis, G.N. (1980) Intelligent Control for Advanced Automated Processes, in *Conf. on Automated Decision Making and Problem Solving,* NASA, Langley.

Saridis, G.N. (1983) , Intelligent Robotic Control, *IEEE Trans. Automatic Contr.,* Vol.AC-28, No.5, pp547-557.

Saridis, G.N. (1985) , Foundations of Intelligent Controls, *Proc.IEEE Workshop on Intelligent Contr.,* pp23-27, RPI., Troy, N.Y.

Saridis, G.N. and Valavnis, K.P. (1988a) , Analytical Design of Intelligent Machines, *IIfac Journal Automatica,* Vol., No.2, pp.

Saridis, G.N. and Moed, M.C. (1988b) Analytical Formulation of Intelligent Machines as Neural Nets, *Proc. 3rd IEEE Int. Symposium on Intelligent Contr.,* Arlington, VA.

Stefik, M. (1981a) Planning with Constraints (MOLGEN: Part 1), *Artificial Intelligence,* Vol.15, No.2, pp111-140.

Stefik, M. (1981b) Planning and Meta-Planning (MOLGEN: Part 2), *Artificial Intelligence,* Vol.15, No.2, pp141-170.

Tate, A. (1977) Generating Project Networks, *Proc. 5th IJCAI,* Cambridge, MA., pp888-893.

Wang, F.-Y., (1988a), *The Condition-Event Net,* in preparation.

Wang, F.-Y. and Saridis, G.N (1988b) A Formal Model for Coordination of Intelligent Machines, *Proc. 3rd IEEE Int. Symposium on Intelligent Contr.,* Arlington, VA.

Warren, D.H.D. (1974) *WARPLAN: A System for Generating Plans,* Memo 76, Dept. Computational Logic, School of AI, University of Edinburgh, Edinburgh, UK.

Wilkins, D.E. (1984) Domain-Independent Planning: Representation and Plan Generation, *Artificial Intelligence,* Vol.22, No.3, pp269-301.

Wilkins, D.E. (1985) Recovering from Execution Errors in SIPE, *Computational Intelligence,* Vol.1, No.1, pp33-45.

Wonham, W.M. and Ramadge, P.J. (1988) Modular Supervisory Control for Discrete Event Systems, Mathematics of Control, Signals, and Systems, Vol.1, No.1 pp13-30.

Yang, C.C (1986) *Relational Databases*, Prentice-Hall, Englewood Cliffs, N.J.

Vaz, A.-F. and Wonham, W.M. (1986), On Supervisor Reduction in Discrete-Event Systems, *Int. J. Control*, Vol.44, No.2, pp475-491.

Zadeh, L.A. (1971), Quantitative Fuzzy Semantics, *Information Sciences*, 3, pp159-176.